WE CLAIM:

1. Apparatus for processing data, said apparatus comprising:

processing logic operable to perform processing operations under control of program instructions; and

an interrupt controller operable:

in response to a first interrupt event, to save to a stack data store first state data associated with processing being performed when said first interrupt event occurred and to redirect program instruction execution to a first interrupt handling program;

upon completion of said first interrupt handling program, to detect if one or more second interrupt events having a higher priority than said processing that was interrupted by said first interrupt event has occurred during execution of said first interrupt handling program and:

(i) if such a said one or more second interrupt events has occurred, then to redirect program instruction execution to a second interrupt handling program without saving further state data to said stack data store; and

(ii) if such a said one or more second interrupt event has not occurred, then to reload said first state data from said stack data store and to resume said processing that was interrupted by said first interrupt event.

2. Apparatus as claimed in claim 1, wherein said first state data includes one or more of:

a program counter value corresponding a current program execution point;

a processor status register value corresponding one or more state variables of said apparatus; and

one or more data processing register values corresponding to data values held within at least some general purpose data processing registers of said apparatus.

3. Apparatus as claimed in claim 1, wherein said processing being performed when said first interrupt event occurred was one of:

execution of a non-interrupt triggered program; and

execution of an active interrupt handling program, said interrupt controller being a nested interrupt controller permitting a pending interrupt handling program to

pre-empt said active interrupt handling program if said active interrupt handling program has a lower priority than said pending interrupt handling program.

4. Apparatus as claimed in claim 1, wherein said first interrupt event and said one or more second interrupt events each have respective priority values, said interrupt controller being operable to compare said respective priority values to determine if any of said one or second interrupts event has a higher priority than said first interrupt event and if so to pre-empt execution of said first interrupt handling program with execution of said a second interrupt handling program.

5. Apparatus as claimed in claim 4, wherein said respective priority values are programmable values.

6. Apparatus as claimed in claim 1, wherein said interrupt controller is responsive to a late interrupt signal during reloading of said first state data to abort a return to said processing being performed when said first interrupt event occurred and instead redirect execution to an interrupt handling program associated with said late interrupt signal.

7. Apparatus as claimed in claim 1, wherein if such a said second interrupt event has occurred, then redirection of program instruction execution to said second interrupt handling program occurs without reloading said first state data from said stack data store.

8. Apparatus as claimed in claim 6, wherein upon aborting said return, said stack data store is repaired to undo any alterations made by partial completion of said return.

9. Apparatus as claimed in claim 8, wherein said repair includes repairing one or more of stack pointer data and link register data.

10. Apparatus as claimed in claim 1, wherein transfer of data values to said stack data store under control of said interrupt controller is performed in parallel with and

asynchronously to loading of program counter location and program instructions into an instruction pipeline prior to execution.

11. Apparatus as claimed in claim 1, wherein said interrupt controller is responsive to execution of a return instruction with a predetermined link address value loaded within a link register to perform a return from interrupt operation.

12. Apparatus as claimed in claim 1, wherein said stack data store is a stack memory.

13. Apparatus as claimed in claim 1, wherein when there are no pending interrupts said apparatus enters a low power mode in which processing is halted awaiting an interrupt event.

14. A method of processing data, said method comprising the steps of:

performing processing operations under control of program instructions; and

in response to a first interrupt event, saving to a stack data store first state data associated with processing being performed when said first interrupt event occurred and to redirect program instruction execution to a first interrupt handling program; wherein

upon completion of said first interrupt handling program, detecting if a second interrupt event having a higher priority than said processing that was interrupted by said first interrupt event has occurred during execution of said first interrupt handling program and:

(i) if such a said second interrupt event has occurred, then redirecting program instruction execution to a second interrupt handling program without saving further state data to said stack data store; and

(ii) if such a said second interrupt event has not occurred, then reloading said first state data from said stack data store and resuming said processing that was interrupted by said first interrupt event.

15. A method as claimed in claim 14, wherein said first state data includes one or more of:

a program counter value corresponding a current program execution point;

a processor status register value corresponding one or more state variables of an apparatus perform said method; and

one or more data processing register values corresponding to data values held within at least some general purpose data processing registers of said apparatus.

16. A method as claimed in claim 14, wherein said processing being performed when said first interrupt event occurred was one of:

execution of a non-interrupt triggered program; and

execution of an active interrupt handling program, interrupt control being nested interrupt control permitting a pending interrupt handling program to pre-empt said active interrupt handling program if said active interrupt handling program has a lower priority than said pending interrupt handling program.

17. A method as claimed in claim 14, wherein said first interrupt event and said second interrupt event each have respective priority values, said respective priority values being compared to determine if said second interrupt event has a higher priority than said first interrupt event.

18. A method as claimed in claim 17, wherein said respective priority values are programmable values.

19. A method as claimed in claim 14, wherein in response to a late interrupt signal during reloading of said first state data, aborting a return to said processing being performed when said first interrupt event occurred and instead redirecting execution to an interrupt handling program associated with said late interrupt signal.

20. A method as claimed in claim 14, wherein if such a said second interrupt event has occurred, then redirection of program instruction execution to said second interrupt handling program occurs without reloading said first state data from said stack data store.

21. A method as claimed in claim 19, wherein upon aborting said return, said stack data store is repaired to undo any alterations made by partial completion of said return.

22. A method as claimed in claim 21, wherein said repair includes repairing one or more of stack pointer data and link register data.

23. A method as claimed in claim 14, wherein transfer of data values to said stack data store is performed in parallel with and asynchronously to loading of program counter location and program instructions into an instruction pipeline prior to execution.

24. A method as claimed in claim 14, wherein in response to execution of a return instruction with a predetermined link address value loaded within a link register, performing a return from interrupt operation.

25. A method as claimed in claim 14, wherein said stack data store is a stack memory.

26. A method as claimed in claim 14, wherein when there are no pending interrupts a low power state is entered in which processing is halted awaiting an interrupt event.